Integrated Fluid and Packet Network Simulations *

George F. Riley[1]
Talal M. Jaafar[1]
Richard M. Fujimoto[2]

[1]College of Engineering
Department of ECE
Georgia Institute of Technology
Atlanta, GA 30332-0250
{riley,jaafar}@ece.gatech.edu

[2]College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
fujimoto@cc.gatech.edu

August 9, 2002

Abstract

A number of methods exist that can be used to create simulation models for measuring the performance of computer networks. The most commonly used method is packet level simulation, which models the detailed behavior of every packet in the network, and results in a highly accurate picture of overall network behavior. A less frequently used, but sometimes more computationally efficient, method is the *fluid model* approach. In this method, aggregations of flows are modeled as fluid flowing through pipes, and queues are modeled as fixed capacity buckets. The buckets are connected via pipes, where the maximum allowable flow rate of fluid in the pipes represents the bandwidth of the communication links being modeled. Fluid models generally result in a less accurate picture of the network's behavior since they rely on aggregation of flows and ignore actions specific to individual flows.

We introduce a new hybrid simulation environment that leverages the strong points of each of these two modeling methods. Our hybrid method uses fluid models to represent aggregations of flows for which less detail is required, and packet models to represent individual flows for which more detail is needed. The result is a computationally efficient simulation model that results in a high level of accuracy and detail in some of the flows, while abstracting away details of other flows. We show a computational speedup of more than twenty in some cases, with little reduction in accuracy of the simulation results.

1 Introduction

Simulation has become the method of choice for many networking research problems. As new protocols are designed and tested, computer based simulations are used to validate the correctness of the new protocol, and are used to measure the performance of the new protocol under a variety of experimental conditions. As new network infrastructure methods are designed (e.g. active queue management methods such as Random Early Detection (RED) [5] and Proportional-Integral (PI) controllers [8]), they are often tested and validated using simulation techniques.

However, as the size and capacity of modern networks have increased, the ability to accurately simulate such networks has decreased. Growth in network bandwidth alone has seen a three order–of–magnitude increase in recent years, placing severe strains on CPU cycles needed to simulate these high speed networks. For example, for a relatively modest OC12 link at 622Mbps, a detailed packet level simulation of a single link can result in over 500,000 simulation events per second. It's easy to see that modeling more than just a few such links can cause the overall execution time of a simulation to become excessively large.

One approach to deal with the increased computational complexity of these simulations is to model the networks in a different way. In [9], Huang proposed a selective abstraction approach whereby some subset of data flows are abstracted to model end–to–end behavior, rather than each individual packet in the network. Several researchers have looked at a *fluid model* approach [12, 11, 27, 21, 22, 17, 10, 15], where the network is modeled as fluid flowing through pipes, rather than individual packets flowing on communication links.

Both of these abstraction techniques can, in some cases, result in a more computationally efficient model. However, these methods usually result in less accurate results due to abstracting away some important details in the behavior of the network. For example, the fluid model assumes that data lost for flows entering a full queue is directly proportional to the inflow rate for each flow entering the queue. While this is certainly true over a long term average, over shorter timescales loss patterns for individual flows may vary dramatically.

In our work, we seek to take advantage of the strong points of both modeling methods (fluid and packets), by creating an integrated hybrid simulation environment. We

*This work is supported in part by NSF under contract number ANI-9977544 and DARPA under contract number N66002-00-1-8934.

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 IEEE

use fluid models to model portions of the network where computational efficiency is more important than precise accuracy, and we use packet–level simulation to model portions of the network where exact packet–by–packet level detail is needed. The result is an integrated simulation environment that can be significantly faster than the packet models, while at the same time sacrificing little in terms of accuracy of results.

The remainder of this paper is organized as follows. Section 2 gives an overview of the fluid models and packet models, and describes our hybrid approach. Section 3 describes in detail our integration effort in creating the hybrid simulation environment. Section 4 describes the experimental methodology we used to validate our hybrid simulator. Section 5 gives the results of the experiments. Finally, section 6 gives some conclusions and future directions of our research.

2 Conceptual Overview

In this section, we discuss the basic simulation methodology used by the fluid model and packet model simulators, highlighting the differences, strengths, and weaknesses of each. Then we show how we integrated the two modeling paradigms into a single simulation, leveraging the strengths of each method. The subsequent sections describe in more detail the integration and the experiments we performed to validate our approach.

2.1 Fluid Models

A simple fluid network model is shown in Figure 1. With the fluid model approach for networks, data flowing between systems are modeled as fluid flowing through pipes. For this discussion, we assume non–looping, feed–forward networks only, but the basic modeling method (with some slight modifications) can be applied for feedback networks as well. At each station in the network model, the flow of data is modeled with six continuous processes, as follows:

1. $\alpha(t)$: the input flow rate (inflow) process into the station. This models the aggregate rate (bits/sec) of data being received at the station. This can be either from the output of a previous station (as in the input to station 3 in the figure), or an external data source model (as in the input to station 1 in the figure).

2. $\beta(t)$: the service rate process, namely, the maximal fluid discharge rate from the server. This models the maximum capacity (bits/sec) of the output communication link from that station.

3. $c(t)$: the buffer capacity process. We point out that it may generally be a function of time, so as to capture the effects of a shared-buffer used by competing flows; however, in our present work we assume it to have a positive fixed value, C bits.

4. $x(t)$: the buffer occupancy process, namely the fluid volume in the buffer. This models the amount of data queued and awaiting transmission on the output link.

5. $\delta(t)$: the fluid discharge rate (outflow) process from the server. This models the rate (bits/sec) of data leaving the station, and (presumably) arriving at a subsequent station (see the input to station 3 in the figure is the sum of the outflow rates at stations 1 and 2). Note that the outflow rate $\delta(t)$ can never be greater than the service rate $\beta(t)$. In other words, a station can never send data to a subsequent station at a rate higher than the maximum bandwidth of the physical communication link connecting the two. The outflow rate can be less than $\beta(t)$ however, since a station with an empty queue will never output more data than is being received.

6. $\gamma(t)$: the loss rate (overflow) process due to a full buffer. This models the data lost at the station.

The first three processes, $\alpha(t)$, $\beta(t)$ and $c(t)$, characterize the behavior of a fluid station, and are referred to as *defining processes*. The other three processes, $x(t)$, $\delta(t)$, and $\gamma(t)$, are determined by the defining processes, and are referred to as *derived processes*. See [25] for a detailed description of basic principles in fluid flow network modeling.

In creating the simulation model using the fluid flow method, at any time instant t, the three derived processes ($x(t)$, $\delta(t)$, and $\gamma(t)$ are calculated precisely for each station, using current values for the defining processes at that station. When a calculated output rate $\delta(t)$ becomes the input rate $\alpha(t)$ for a subsequent station, the derived processes for that station are calculated, and the process repeats until all stations with changes to the defining processes have calculated the derived processes. At this point, the network has reached a *steady state*, and no further computation is needed until one of three things occurs:

1. One of the external data source models indicates a new data generation rate. For example, an on–off data source model indicates that the source has changed from on to off, or from off to on.

2. A queue transitions from non–empty to empty (i.e. $x(t) = 0$ at a given station). This potentially results in a changed inflow rate at a downstream station.

3. A queue transitions from nearly full to full (i.e. $x(t) = c$ at a given station). This results in a change in the overflow rate at the station.

Upon occurrence of any of the above three events, the derived processes for each station are again calculated as above, and the network again reaches steady state.

A number of trade-offs must be considered when choosing to model a network in this fashion.

1. The simulation of the fluid model can be, in some cases, extremely computationally efficient. Since the simulation only models *rate change* events, the total computational complexity of the simulation can be much less than the packet level simulation models. We have shown in prior experiments a speedup of one hundred or more when modeling simple networks with the fluid approach as compared to detailed packet level simulations. However, this computational efficiency becomes less pronounced as the

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 IEEE

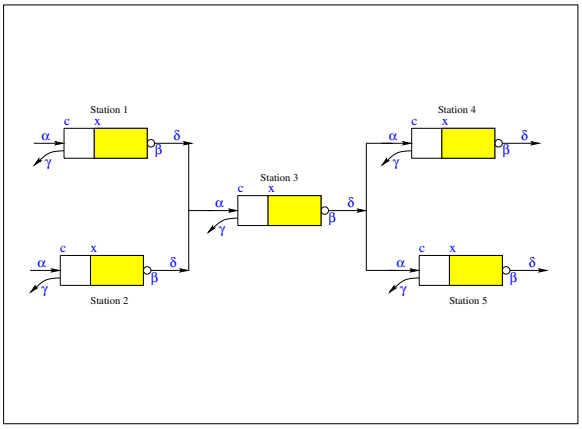

Figure 1: Simple Fluid Model

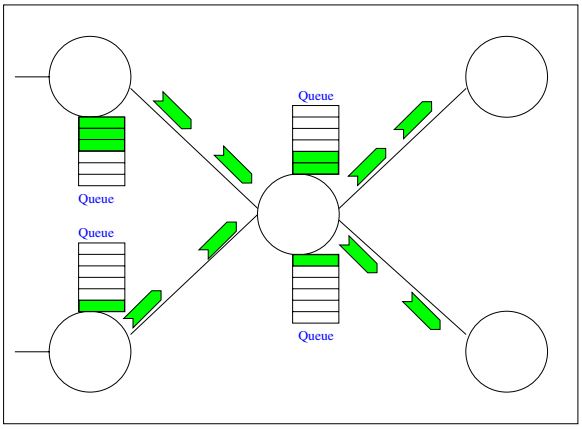

Figure 2: Simple Packet Model

size of the network being modeled grows. This is partially due to the well–known *Ripple Effect* in fluid simulations. The ripple effect is caused by an input flow rate change in a non–empty queue, which causes output flow rate changes for all flows sharing the queue. Thus a single rate change event for one flow can cause multiple rate change events in downstream queues. See Liu et al. [12, 11] for a detailed discussion the ripple effect.

2. The fluid modeling method can be more susceptible to mathematical analysis than other modeling methods. Wardi et al. [24] have shown that describing the network behavior as a set of differential equations can lead to accurate *Infinitesimal Perturbation Analysis* of network behavior, giving good predictions for future network performance. Towsley et al. [16] used a fluid model and differential equations approach to analytically describe the behavior of TCP endpoints and active queue management techniques.

3. The fluid modeling method often results in reduced accuracy in the network performance metrics produced by the simulations. The fluid models always assume average behavior over some time horizon, and do not necessarily capture the effects of unlucky or unusual network behavior. For example, when several flows are arriving at a full queue, the fluid model computes loss rates for each flow proportional to their input rate. In actual networks, the loss rates of each of the competing flows can vary widely at any point in time, due to simply bad luck, or stroboscopic effects of the arrival patterns.

4. As previously mentioned, networks with feedback (either with potential routing loops in the network topology or with input sources that adjust inflow rates as a function of network behavior) are more difficult to simulate using fluid models. Routing loops can cause an explosion in the number of rate change events processed by the fluid model simulator, vastly reducing the efficiency gains that are achievable.

Despite these tradeoffs and potential loss of accuracy, a number of researchers have reported good results by using the fluid flow models for network simulation [27, 21, 22, 17, 10, 15].

2.2 Packet Models

A sample packet level simulation is shown in Figure 2. In this method of modeling networks, the simulator retains information about every packet generated at each source, and models the path of the packet at every point in the network. Packets are tracked individually on each link, in each queue, and at each data source and sink. Packet losses are computed deterministically on a packet by packet basis, leading to fairly precise and repeatable network behavior. A number of research efforts and commercial products exist that utilize the packet modeling method for network simulation, including *ns2* [14], *pdns* [19, 20], *GloMoSim* [28], *SSF* [4, 3], *JavaSim* [23], and *OpNet* [1].

The packet level modeling method has the drawback of increased computational complexity. Since the simulator tracks and models every packet event (enqueue, dequeue, transmit, receive, loss) for every packet in the system, the total number of events processed can become excessively large. Additionally, computer memory requirements also grow in direct proportion to the total number of simulated packets being modeled at any point in time, contributing to memory resource limitations in packet level simulations.

2.3 The Hybrid Approach

Our hybrid approach to network simulation leverages the strong points of both the fluid and packet modeling methods, giving the computational efficiency of fluid models combined with the accuracy of packet models. We create an integrated environment where a portion of the data flowing in the network is modeled using the fluid method, and the remainder is modeled using the packet method. A sample network topology illustrating how the fluid and packet models are combined to create the hybrid model is shown in Figure 3.

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 IEEE

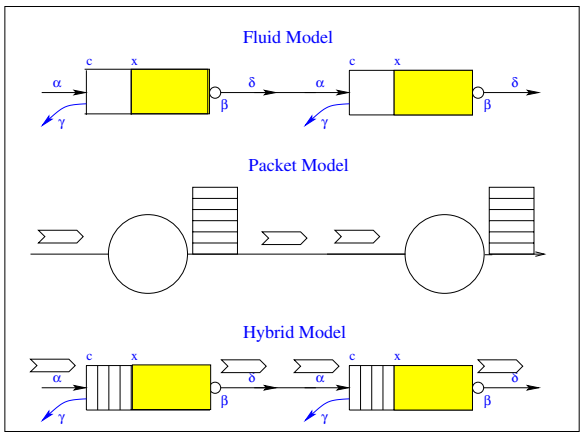

Figure 3: Hybrid Model

We start with the assumption that the network simulation consists of two different classes of data flows. First is a set of *foreground* traffic flows, for which detailed and accurate modeling is needed. Second is a set of *background* flows, which exist and are modeled to compete with the foreground flows, but for which detailed models are not needed. The background flows exist to provide realistic models of competing traffic for the foreground flows, to allow the foreground flows to encounter realistic congestion patterns at the various routers and links in the network.

Using our integrated environment, which is described in detail in the next section, we create two simulations. Each simulator (fluid or packet–level) models the entire network, but only one class of traffic. We use the fluid modeling approach to simulate the background flows, using the existing (but slightly modified) *Hybrid Discrete-Continuous Fluid-Network Simulator (HDCF-NS)* [15] fluid simulator. We use the packet modeling approach to simulate the foreground flows, using the existing (but again slightly modified) *pdns* [20] packet simulator. Of course, both simulations model the same network topology, with the same queue limits and link capacities at each station. The simulators then exchange information at run–time, allowing the packet model simulation to become aware of buffer occupancy due to the fluid model flows.

The *pdns* simulator was enhanced to include a model called a *FluidQueue*. This queue is a *DropTail* queue with fixed capacity, but with the addition of a fluid level indicator and a level rate change indicator. When the fluid model detects a change in the buffer occupancy function $x(t)$ (described previously), it informs the corresponding FluidQueue model in the *pdns* simulation of the new fluid level rate change, using a distributed simulation technique described in the next section. These rate change events occur when an input rate changes, or when the buffer becomes full or empty.

For the actions needed when a rate change event is recieved, or when a packet in the packet model simulation arrives at a FluidQueue, we first define the follwing variables. Subscripts k and f refer to the information from packet simulator and fluid simulator respectively.

T_k = Time of Rate Change Event k

x_k = Fluid Level at time T_k

x'_k = Rate of Change of X_k

T_f = Time of Fluid Rate Change Event

x'_f = Rate of Change of X from Fluid at time T_f

S_p = Size of arriving packet

C = Maximum buffer capacity

β = Link Speed of Output Link

D_p = Delay of arriving packet due to fluid

T_p = Time delay for pending packet to fit in the queue (described below)

When a rate change event from the fluid simulator is received at time T_f, the following actions are taken.

1. Calculate the new fluid level $x_{k+1} = x_k + (T_f - T_k)x'_k$.
2. Set the new fluid level rate change $x'_{k+1} = x'_f$
3. Set time of last update $T_{k+1} = T_f$

When a packet arrives at the FluidQueue in the packet level simulation at time T_{k+1}, the following actions are taken.

1. Calculate the new fluid level $x_{k+1} = x_k + (T_{k+1} - T_k)x'_k$. Note that x'_k is constant in the interval $[T_k, T_{k+1}]$ since any rate changes would be processed as above, resulting in an updated T_k.
2. Calculate the extra queuing delay for this packet due to the fluid in the queue, $D_p = x_{k+1}\beta$. In other words, the packet is assigned additional delay due to fluid in the queue ahead of the arriving packet.
3. If the $x_{k+1} + S_p \leq C$, admit the packet to the queue.
4. If the $x_{k+1} + S_p > C$, and if there is no *pending packet* (see step 5 below), admit the packet with probability $p = 1 - (\alpha(t) - \beta(t))/\alpha(t)$. In other words, accept the packet with a probability proportional to the fraction of the arriving fluid that is lost. If there is a pending packet, drop the arriving packet unconditionally.
5. If a packet is accepted probabalistically in step 4 above, this packet becomes a *pending packet*. Compute the amount of time it takes for the pending packet to fit completely in the queue $T_p = (x_{k+1} + S_p - C)\ \beta$. In other words, determine when excess data (above the specified queue limit C) will fit in the queue, based on the amount of excess data and the queue's outflow rate. Schedule a future event at time $T_{k+1} + T_p$ to indicate the packet is no longer pending and is now fully resident in the queue.

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 IEEE

The probabilistic acceptance of packets into nearly full fluid queues, and the processing of the pending packet are needed to realistically model the behavior of packets arriving at congested queues. Even when packets arrive at completely full queues (in real packet networks), there is still a non-zero probability of acceptance, based on the output rate of packets, and the relative input rate if incoming packets. The above algorithm models this behavior.

3 Integration Methodology

As previously mentioned, we created an integrated simulation environment that incorporates both the fluid models and the packet models, simulating the same network at the same time. However, we wanted to leverage existing simulation technology and use existing simulators with as little modification as possible. For the fluid model simulator, we chose the *HDCF-NS*[15] simulator, which is written in Java. For the packet model simulator, we chose *parallel and distributed ns* (*pdns*)[20], which is written in C++ and *TCL*. We point out that the *HDCF-NS* simulator already has built in a rudimentary packet and fluid hybrid model. But, the *HDCF-NS* packet endpoint models lack support for any layer–4 protocol. For our work we preferred to utilize the full–featured packet level simulation environment found in *ns2* and *pdns*. The two simulators run autonomously (i.e. in separate processes and separate address spaces), and exchange information at runtime about the state of the simulated network. In all of our experiments, the two processes execute on separate workstations connected by a Giga-Bit Ethernet network.

In such a distributed simulation environment, the simulators need to coordinate with each other in two basic ways. First they need *time management* mechanisms to insure that the simulation time between the simulators advance in a manner that insures proper event causality. Second is *message exchange* mechanisms, that the simulators can use to inform peers about state changes in simulated objects. To facilitate this interaction, we utilized the existing Georgia Tech *Runtime Infrastructure Kit* (*RTIKIT*) [6, 7] originally developed by Fujimoto. The RTIKIT is designed specifically to provide these services to distributed simulation processes. The RTIKIT is part of a larger distributed network simulation support environment called the *Dynamic Simulation Backplane* [18, 26]. The backplane is designed specifically to facilitate the exchange of network protocol packets between heterogeneous simulators.

Conceptually, the fluid model simulation must inform the packet model simulation of changes in the fluid level at the queues. Furthermore, the two simulators must take steps to insure the simulation time values stay in agreement as the simulation progresses. To facilitate these steps, we modified the *HDCF-NS* simulator in two simple ways.

1. The main event processing loop was modified to specifically request permission to advance simulation time to the time of the next event. Existing functionality in the RTIKIT provides a distributed consensus algorithm to insure that simulation time is advanced in a manner that insures correct event causality between the simulators. Using this functionality, the time management capability was added to *HDCF-NS* with just a few lines of C code linked in using the Java Native Interface (JNI) package.

2. Each time a change in the fluid level rate of change $x'(t)$ is detected by *HDCF-NS*, the RTIKIT message exchange mechanisms are invoked to notify *pdns* that the fluid level at the queue is changing at a different rate. The message sent to *pdns* includes the simulation time, the queue number (a unique integer value agreed to by both *HDCF-NS* and *pdns*), the fluid level, the rate of change of fluid level, the input rate, and the output rate. Existing functionality in the RTIKIT provides this message passing functionality, and again was achieved with just a few lines of C code using JNI.

Since the *pdns* packet level simulator is already designed to operate in a distributed simulation environment, it already contains time management functionality in the main event processing loop. The changes to *pdns* required for the hybrid model simulations were limited to the addition of the *FluidQueue* element previously discussed, and some code in the message processing routines to detect and forward the fluid rate change messages discussed above.

4 Experimental Methodology

To illustrate the effectiveness of our hybrid approach, we chose a simulation of a well known web response time experiment by Jeffay et al. [2]. The experiment is designed to measure the response time of individual web objects under a variety of conditions. The topology for the original experiment is shown in Figure 4. Each of the web browser systems models the behavior of several hundred simultaneous web browsing sessions. The size of individual web object requests and the time delay between the requests is modeled based on empirical measurements described by Mah in [13]. Each web server system models a single web server, each handling a large number of simultaneous requests.

Such an experiment seemed to us to be an ideal platform for demonstrating the foreground and background traffic models, and the hybrid simulation approach. We modified the original topology slightly (as shown in Figure 5), to increase the available bandwidth on the bottleneck link, while at the same time adding background traffic sharing the link and competing with the foreground web browser models. In this case the web object requests and responses are the foreground traffic (which is being measured by the experiment). The background traffic is the competing traffic which is not being measured but exists to provide time–varying congestion levels to the foreground flows. For this experiment, we used bursty on–off data sources with a pareto distribution of on–off times as the competing traffic. The experiment was performed with various levels of foreground and background traffic intensity to measure effectiveness of the hybrid approach under different conditions.

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 IEEE

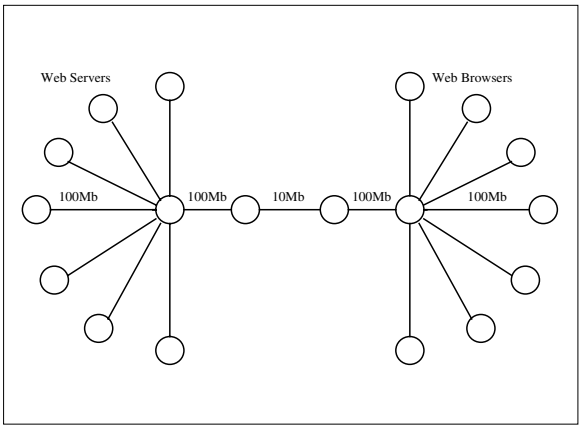

Figure 4: Original Web Browsing Topology

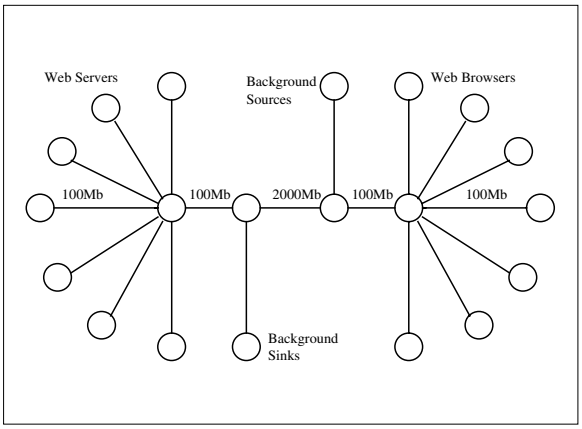

Figure 5: Modified Web Browsing Topology

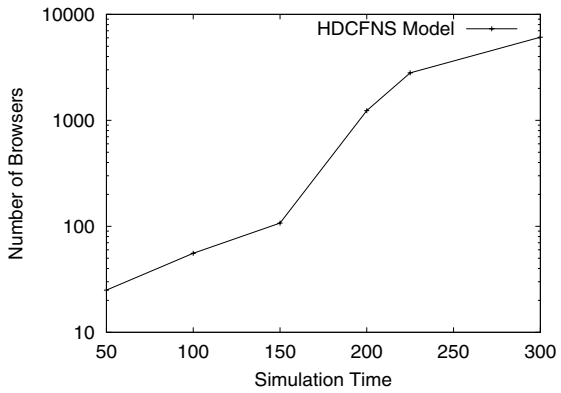

Figure 6: HDCFNS Behavior

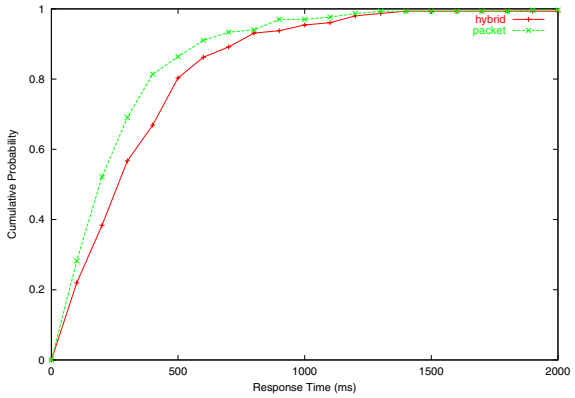

Figure 7: WRT with 50 Background Flows

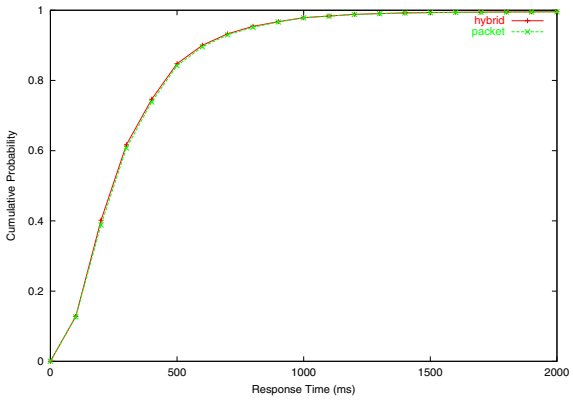

Figure 8: WRT with 150 Background Flows

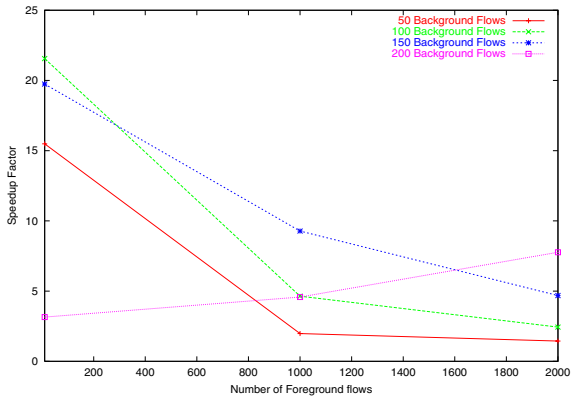

Figure 9: Speed Factor of the Integrated Model

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 **IEEE**

We performed two basic sets of experiments: baseline and integrated. The baseline experiment uses *pdns* packet level simulator to model all the traffic flows, web browsers, web servers, and competing traffic. As previously described, *pdns* models precisely all packets in the network, resulting in good accuracy, at the expense of longer simulation time. The length of simulation, and the measured web object response time provide the baseline with which to compare the integrated approach. The integrated experiment uses the *HDCF-NS* fluid model simulator to simulate the competing traffic, and *pdns* to simulate the web traffic. Although the fluid model is an approximation, we found it to be acceptable in this application since the background traffic is not being measured. The background traffic exists to cause varying congestion levels at the bottleneck of the topology. The integrated fluid packet model is expected to perform better than the baseline approach in terms of execution time while maintaining comparable accuracy in the results. The Web Response Time (WRT) cumulative distribution function (CDF) was used in the experiments as the metric for determining the accuracy of the simulation results.

5 Experimental Results

We noticed early in our experiments the *ripple effect* for fluid simulation models. While the fluid model simulator performed extremely well modeling moderate numbers of flows (up to about 150), the execution time increased dramatically as the number of flows increased over approximately 200. This can be seen in Figure 6. Thus the hybrid simulation approach looses effectiveness at these scales.

As mentioned earlier, we used web object response time as the metric for comparing the accuracy of the hybrid approach. The web response time is measured by noting the time of each web object request and the time each object is completely delivered to the browser. Figure 7 and Figure 8 shows the comparison of web response time between both integrated and base (pdns) models. Figure 7 is the result of a network topology with 50 competing flows and 10 browsers. It is easy to see in the CDF that 85 percent of the web objects got served at at 500 seconds in this experiment. Figure 8 is the result of a network topology with 150 competing flows and 1000 browsers, where 80 percent of the web object got served at 500 seconds. This is expected since there are more traffic flows in the network which might cause delays in serving the web requests. However, the results show clearly there is little difference in the accuracy of the measured foreground traffic behavior in the hybrid approach, as compared to the packets only approach.

In addition,the goal of this new integrated approach is to speedup the simulation while maintaining the results accuracy. Figure 9 shows the performance of the integrated model comparing to the baseline. As illustrated, speedup factors ranging from 15 to 21 were obtained when there are a small number of browsers, decreasing to about 2 to 4 as the number of browsers reach 2000. The decrease in speedup is expected since the number of foreground flows are increasing relative to a fixed amount of background traffic.

We also point out that there is some overhead with the synchronization of the hybrid and packet models in the integration, and that can be seen when there is a low number of browsers and low number of competing flows. In this case, the overhead of message passing and synchronization largely offsets the gain from the hybrid approach.

Additionally, we see clearly the ripple effect in the fluid simulation in our speedup graphs. For the case with 200 background flows, the execution time of the fluid simulator becomes the pacing factor for the hybrid simulation, resulting in the relatively poor speedup numbers in this case. However, for cases with moderate numbers of competing flows, (less than 200), we see substantial speedup as expected.

6 Conclusions and Future Work

In this paper, we described the integrated fluid/packet model, and evaluated its performance over the traditional packet-level model. The web response time was used as a measure key for comparing the accuracy of the simulation results, while the simulation execution time was the basic factor of simulation performance. Multiple simulations for different network scenarios were analyzed to calculate the relative performance of the new model. As expected, the integrated model takes advantage of the computational efficiency of the fluid simulation, and the details of interest measures of the packet-level simulation. However, the performance of the integrated model in more complex networks (more traffic flows) was bounded by the ripple effect of the fluid simulation.

Furthermore, our current integrated fluid/packet model does not implement the effects of foreground traffic on the background (i.e. the exchange of information is one-way). We are presently working on ways to efficiently adjust fluid flow rates and levels in response to packet level events in the packet based simulation.

References

[1] S. Bertolotti and L. Dunand. Opnet 2.4: an environment for communication network modeling and simulation. In *Proceedings of the European Simulation Symposium*, October 1993.

[2] Mikkel Christiansen, Kevin Jeffay, David Ott, and F. Donelson Smith. Tuning RED for web traffic. In *Proceedings of ACM SIGCOMM 2000*, pages 139–150, Aug 2000.

[3] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulations. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 1999.

[4] J. H. Cowie, D. M. Nicol, and A. T. Ogielski. Modeling the global internet. *Computing in Science and Engineering*, January 1999.

[5] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE Transactions on Networking*, 1(4):397–413, August 1993.

[6] Richard M. Fujimoto, Kalyan Permualla, and Ivan Tacic. Design of high performance RTI software. In *Distributed Simulation and Real-Time Applications 2000*, August 2000.

[7] R.M. Fujimoto, S. Ferenci, M. Loper, T. McLean, K.S. Perumalla, G.F. Riley, and I Tacic. Fdk users guide. Georgia Institute of Technology, March 2001.

[8] C.V. Holloty, V. Misra, D. Towsley, and W. B. Gong. Analysis and design of controllers for aqm routers supporting tcp flows. *IEEE Transactions on Automatic Control (to appear)*, 2002.

[9] P. Huang, D. Estrin, and J. Heideman. Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols. In *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, July 1998.

[10] G. Kesidis, A. Singh, D. Cheung, and W.W. Kwok. Feasibility of fluid–driven simulation for atm networks. In *Proceedings of IEEE GLOBECOM 96*, Nov 1996.

[11] B. Liu, Y. Guo, J. Kurose, D. Towsley, and W. Gong. Fluid simulation of large scale networks: issues and tradeoffs. In *Proceedings of PDPTA'99*, June 1999.

[12] Benyuan Liu, Daniel R. Figueiredo, Yang Guo, Jim Kurose, and Don Towsley. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, April 2001.

[13] Bruce A. Mah. An empirical model of http network traffic. In *Proceedings of IEEE INFOCOMM*, pages 592–600, 1997.

[14] S. McCanne and S. Floyd. The LBNL network simulator. Software on-line: http://www.isi.edu/nsnam, 1997. Lawrence Berkeley Laboratory.

[15] Benjamin Melamed, Shuo Pan, and Yorai Wardi. Hybrid discrete-continuous fluid-flow simulation. In *Proc. of the SPIE International Symposium on Information Technologies and Communications (ITCOM 01)*, Aug 2001.

[16] Vishal Misra, Wei-Bo Gong, and Don Towsley. A fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. In *Proceedings of ACM SIGCOMM*, Sep 2000.

[17] David Nicol, Michael Goldsby, and Michael Johnson. Fluid-based simulation of communication networks using ssf. In *Proceedings of 1999 European Simulation Symposium*, June 1999.

[18] George F. Riley, Mostafa H. Ammar, Richard M. Fujimoto, D. Xu, and K. Perumalla. Distributed network simulations using the dynamic simulation backplane. In *Proceedings of the 21st Annual Conference on Distributed Computing Systems*, April 2001.

[19] George F. Riley, Richard M. Fujimoto, and Mostafa H. Ammar. A generic framework for parallelization of network simulations. In *Proceedings of Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'99)*, October 1999.

[20] George F. Riley, Richard M. Fujimoto, and Mostafa H. Ammar. Parallel/Distributed ns. Software on-line: www.cc.gatech.edu/ computing/ compass/ pdns/ index.html, 2000. Georgia Institute of Technology.

[21] D. Ros and R. Marie. Estimation of end-to-end delay in high-speed networks by means of fluid model simulations. In *Proceedings of 13th European Simulation Multiconference*, June 1999.

[22] D. Ros and R. Marie. Loss characterization in high-speed networks through simulation of fluid models. In *Proceedings of SPECTS'99*, July 1999.

[23] Jung-Ying Tyan and Chao-Ju Hou. Javasim: A component-based compositional network simulation environment. In *Proceedings of the Western Simulation Multiconference, Communication Networks And Distributed Systems Modeling And Simulation*, Jan 2001.

[24] Y. Wardi and G. Riley. Ipa for loss volume and buffer workload in tandem sfm networks. In *Proceedings of 6th Workshop on Discrete Event Systems (WODES'02) (to appear)*, Oct 2002.

[25] Yorai Wardi and Benjamin Melamed. Continuous flow models: modeling, simulation and continuity properties. In *Proceedings of 38th IEEE Conference on Decision and Control*, volume 1, pages 34–39, 1999.

[26] Donghua Xu, George F. Riley, Mostafa H. Ammar, and Richard M. Fujimoto. Split protocol stack network simulations using the dynamic simulation backplane. In *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2001.

[27] Anlu Yan and Wei-Bo Gong. Time–driven fluid simulation for high-speed networks. *IEEE Transactions on Information Theory*, 45(5):1588–1599, July 1999.

[28] X Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simlations*, May 1998.

Proceedings of the 10th IEEE Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems (MASCOTS'02)
1526-7539/02 $17.00 © 2002 **IEEE**